Gerenciamento de Pacotes

Conceitos Gerais

1. Introdução

- O Linux possui alguns **Gerenciamento de pacotes** responsável por instalar, atualizar e remover programas. Além disso, eles auxiliam no controle de dependências, uma tarefa que pode ser bastante trabalhosa em alguns casos.

- Muitas vezes, para que um programa funciona corretamente, antes de sua instalação, é necessário instalar uma série de outros programas (bibliotecas).

- Neste ponto, o gerenciador de pacotes torna-se uma ferramenta importante, pois já instala automaticamente tudo que é necessário para o funcionamento do software.

2. Repositório

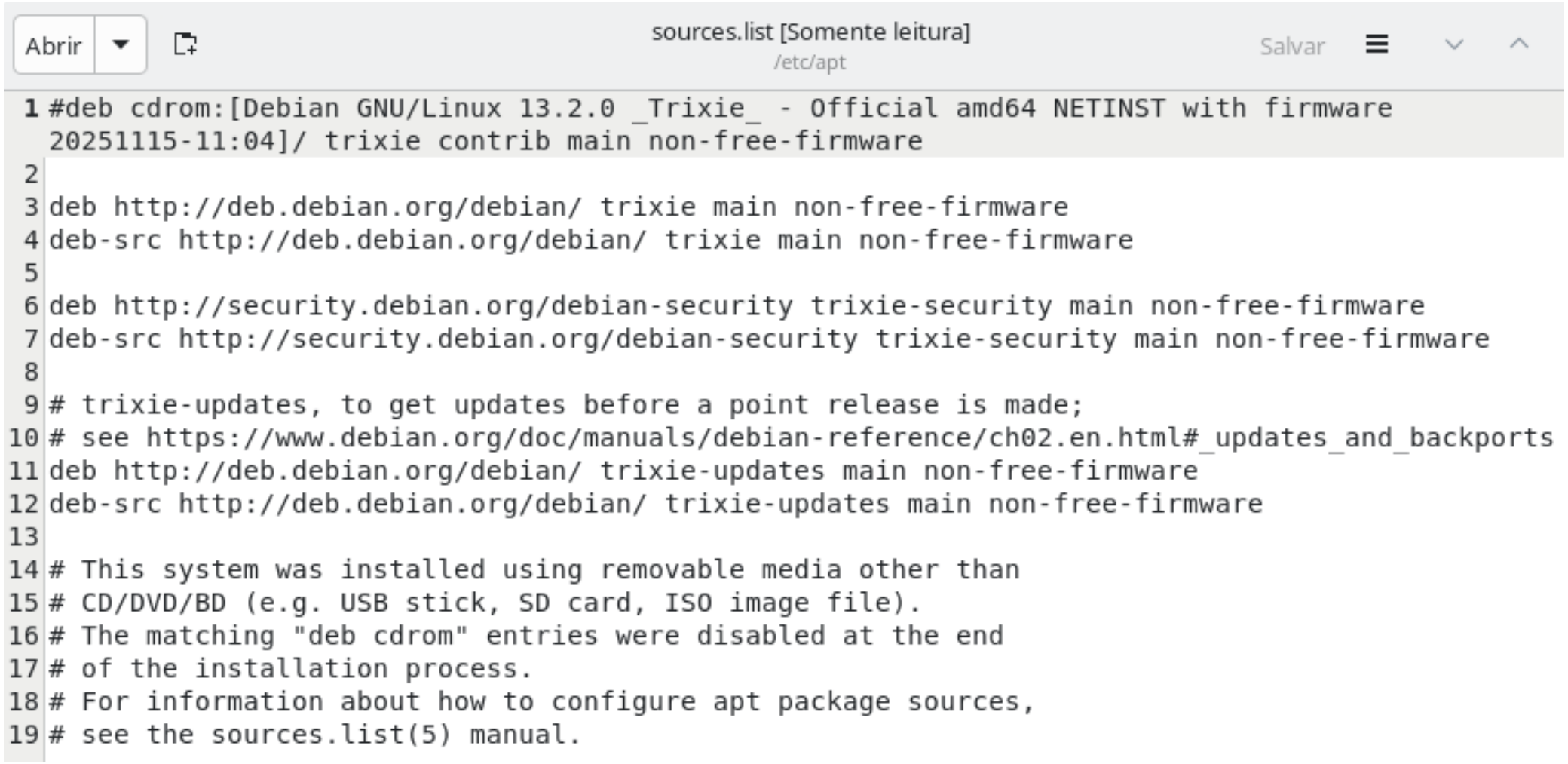
- Um **repositório** é um **servidor (ou conjunto de servidores)** que armazena pacotes de software organizados. Em outras palavras, é como uma **“loja oficial de programas”** do sistema Linux.

- Na prática, quando é dado um comando para instalação, o Linux consulta os repositórios configurados, procura o pacote, baixa da internet e instala automaticamente. O quadro abaixo mostra o arquivo onde é possível consultar e gerenciar os repositórios:

```
gedit /etc/apt/sources.list
```

2. Repositório

- A imagem abaixo mostra o editor aberto com o arquivo "sources.list":



3. APT

- O APT é uma ferramenta desenvolvida originalmente para o Debian (e seus derivados). ELa gerencia pacotes ".deb".

- O seu funcionamento é simples, ele baixa um pacote .deb e suas dependências, e aplica o comando DPKG (comando de instalação) para instalar o pacote.

- Para fazer a instalação de um pacote, basta digitar o coando do quadro abaixo:

```
apt install nome_do_pacote
```

3. APT

- A tabela abaixo mostra os principais comandos do APT:

Comando	Descrição
`apt update`	Atualiza a lista de pacotes disponíveis nos repositórios
`apt upgrade`	Atualiza os pacotes instalados (sem remover nada)
`apt full-upgrade`	Atualiza tudo, podendo instalar/remover pacotes para resolver dependências
`apt install pacote`	Instala um novo pacote
`apt remove pacote`	Remove um pacote, mas mantém arquivos de configuração
`apt purge pacote`	Remove o pacote e seus arquivos de configuração
`apt autoremove`	Remove dependências que não são mais necessárias
`apt search nome`	Procura pacotes pelo nome ou descrição
`apt show pacote`	Mostra informações detalhadas sobre um pacote

3. APT

- O APT pode ser usado com alguns parâmetros, como mostrado na tabela abaixo:

Parâmetro	Nome/Significado	O que faz	Exemplo de uso
`-d`	`--download-only`	Baixa os pacotes, mas **não instala**. Útil para baixar antes ou usar offline depois.	`apt install -d nginx`
`-y`	`--yes`	Responde automaticamente **"sim" para todas as perguntas** durante a instalação/remoção. Muito usado em scripts.	`apt install -y nginx`
`-f`	`--fix-broken`	Tenta **corrigir dependências quebradas** no sistema (instala ou remove pacotes necessários para resolver conflitos).	`apt install -f`

3. APT

- Como exemplo de utilização do APT, vamos instalar o famoso software de edição de imagem GIMP no Linux:

```
#atualiza a lista de pacotes
sudo apt update

#instala o gimp
sudo apt install gimp

#havendo problemas de dependência, uso esse comando
sudo apt install -f
```

4. Update e Upgrade

- Sempre antes de instalar algum programa, é interessante atualizar a lista de pacotes para que a última versão estável seja utilizada.

- O update atualiza o catálogo de programas, ou seja, baixa **informações atualizadas** dos repositórios (atualiza a lista de versões disponíveis dos pacotes). Ele **não instala nada. Em resumo,** só "consulta o catálogo" para ver o que tem de novo.

- Já o upgrade atualiza os programas já instalados para versões mais novas, usando como base a lista que foi obtida com o `update`. **Não remove pacotes nem instala novos automaticamente.**

```
#Exemplo
#atualiza o catalógo e descobre uma nova versão do Gimp
sudo apt update
#atualiza o firefox instalado na máquina
sudo apt upgrade
```

5. Repositórios

- Os repositórios são servidores ao redor do mundo (espelhados para garantir disponibilidade) que disponibilizam os programas do Linux.

- Na maioria das distribuições, o arquivo com o endereço dos repositórios fica em "/etc/apt/sources.list".

- É possível adicionar novos repositórios neste arquivo, aumento o número de programas que podem ser baixados.

- Entretanto, é preciso ter muito cuidado com a adição de repositórios no Linux, principalmente em servidores que estão em produção.

5. Repositórios

- A imagem abaixo mostra o arquivo "/etc/apt/sources.list":

```
1  #deb cdrom:[Debian GNU/Linux 13.2.0 _Trixie_ - Official amd64 NETINST with firmware
   20251115-11:04]/ trixie contrib main non-free-firmware
2
3  deb http://deb.debian.org/debian/ trixie main non-free-firmware
4  deb-src http://deb.debian.org/debian/ trixie main non-free-firmware
5
6  deb http://security.debian.org/debian-security trixie-security main non-free-firmware
7  deb-src http://security.debian.org/debian-security trixie-security main non-free-firmware
8
9  # trixie-updates, to get updates before a point release is made;
10 # see https://www.debian.org/doc/manuals/debian-reference/ch02.en.html#_updates_and_backports
11 deb http://deb.debian.org/debian/ trixie-updates main non-free-firmware
12 deb-src http://deb.debian.org/debian/ trixie-updates main non-free-firmware
```

5. Repositórios

- A imagem abaixo descreve alguns componentes do arquivo "/etc/apt/sources.list":

🟢 `main`
- Software livre (oficial)
- 100% suportado pelo Debian

🟡 `contrib`
- Software livre que depende de algo não livre

🔴 `non-free`
- Software proprietário

🔵 `non-free-firmware`
- Firmwares não livres (Wi-Fi, GPU, etc.)

`deb ... trixie`	Pacotes principais
`deb-src ...`	Código-fonte
`trixie-security`	🔐 Correções de segurança
`trixie-updates`	🔄 Atualizações menores

5. Repositórios

- Eventualmente, podemos precisar de um programa que não está no repositório oficial da distribuição. Neste caso, podemos adicionar.

- Como exemplo, vamos instalar o sistema gerenciador de banco de dados MySQL (da Oracle) no Debian 13. Por padrão, ele não consta no repositório oficial, será necessário adicioná-lo.

- Não é uma boa prática adicionar um repositório direto no "sources.list" do Linux, pois pode ficar desorganizado. O ideal é criar um arquivo em "**/etc/apt/sources.list.d**" com um nome relacionado.

- Desta forma, se necessário apagar o repositório, não existe risco de corromper o apagar por engano um repositório importante do Linux.

5. Repositórios

- O quadro abaixo mostra o passo a passo de como adicionar o repositório do MySQL no Debian 13:

```
#cria um arquivo para ser o repositório
sudo nano /etc/apt/sources.list.d/mysql.list

#adiciona esta linha no arquivo mysql.list
deb http://repo.mysql.com/apt/debian/ bookworm mysql-8.0

#baixa a chave GPG e adiciona no APT
wget https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
sudo apt-key add RPM-GPG-KEY-mysql-2022

#atualiza o catalogo
sudo apt update
```

6. Instalando Programas com DPKG

- O **dpkg** é o **gerenciador de pacotes de baixo nível** das distribuições baseadas em Debian (como Debian e Ubuntu). Ele trabalha diretamente com arquivos "`.deb`", sendo a base para ferramentas como o `apt`.

- Na verdade, o funcionamento do APT consiste em baixar os pacotes (programa e suas dependências), que geralmente são ".deb", e instalar usando o DPKG.

- Pode acontecer de determinados programas não estarem presentes em nenhum repositório do Debian/Ubuntu. Mas o site oficial oferece o programa com a extensão ".deb". Nesse caso, basta fazer o download e instalar utilizando o DPKG.

#Exem

6. Instalando Programas com DPKG

- O dpkg é responsável por instalar, remover e listar pacotes ".deb". Mas é importante reforçar que ele **não resolve dependências automaticamente** (diferente do `apt`). Faltando uma dependência, é necessário resolver manualmente. A imagem abaixo mostra os principais parâmetros do DPKG:

Parâmetro	Função	Exemplo
`-i`	Instalar um pacote `.deb`	`dpkg -i pacote.deb`
`-r`	Remover pacote (mantém configs)	`dpkg -r nome_pacote`
`-P`	Remover pacote completamente	`dpkg -P nome_pacote`
`-l`	Listar pacotes instalados	`dpkg -l`
`-L`	Listar arquivos de um pacote	`dpkg -L nome_pacote`
`-s`	Mostrar informações do pacote	`dpkg -s nome_pacote`

6. Instalando Programas com DPKG

- O quadro abaixo mostra alguns exemplos do uso do dpkg:

```
#instala o chrome manualmente
sudo dpkg -i google-chrome.deb

#se der erro de dependencia, usa esse comando
sudo apt -f install

#lista os pacotes instalados que tenham mysql no nome
dpkg -l | grep mysql
```

7. Removendo Programas com DPKG

- Como exemplo, vamos desinstalar os pacotes do MySQL usando o dpkg:

```
#verifica os pacotes instalados
dpkg -l | grep mysql

#agora remove os pacotes desejados
sudo dpkg -P mysql-server mysql-client mysql-server-8.0

#mesmo com dpkg é bom limpar eventuais dependenciais
sudo apt -f install
sudo apt autoremove
```

8. Verificando Informações do Pacote com DPKG

- O comando abaixo mostra como ver informações de um ".deb" antes mesmo de instalar:

```
#verifica os dados do .deb do chrome
dpkg -I google-chrome.deb
```

```
Package: google-chrome-stable
Version: 122.0.0
Architecture: amd64
Maintainer: Google Inc.
Depends: libc6, libgtk-3-0, libx11-6
Description: Google Chrome web browser
```

9. Instalando Software com APT e com DPKG

- Alguns fabricantes de software oferecem um programa ".deb" que adiciona nos repositórios do APT todos os pacotes e dependências mais atuais para instalação do software. Neste caso, usamos o dpkg e depois o apt para instalação.

- Como exemplo, vamos usar esta técnica para instalar o MySQL:

```
#baixa o pacote diretamente do site ou pelo comando
wget https://dev.mysql.com/get/mysql-apt-config_0.8.29-1_all.deb

#instala para adicionar os repositorios do mysql no APT
sudo dpkg -i mysql-apt-config_0.8.29-1_all.deb

#Atualiza e instala
sudo apt update
sudo apt install mysql-server
```

10. Snap

- O **Snap** é um sistema de empacotamento e distribuição de software criado pela Canonical. Ele permite instalar aplicativos de forma isolada do sistema, com todas as dependências incluídas no próprio pacote.

- Um pacote **Snap** é como um “pacote fechado” que já vem com o programa, bibliotecas necessárias e dependências. Ele roda dentro de um ambiente isolado (sandbox).

- Enquanto no APT, o sistema instala dependências separadamente, o Snap já vem com tudo que for necessário para o programa rodar dentro do pacote.

- Des ta forma, o snap serve para Instalar aplicativos mais facilmente (sem conflito de dependências), ter versões mais atualizadas de programas, rodar apps isolados do sistema (mais segurança) e possibilidade de usar o mesmo pacote em várias distros Linux

10. Snap

- A imagem abaixo traça um comparativo entre APT e Snap:

Característica	Snap	APT
📦 Instalação	Pacote completo (tudo junto)	Depende de bibliotecas do sistema
🔄 Atualizações	Automáticas	Controladas pelo usuário
🧩 Dependências	Inclusas no pacote	Compartilhadas no sistema
⚡ Desempenho	Mais lento para abrir	Mais rápido
💾 Espaço em disco	Usa mais espaço	Mais leve
🔒 Segurança	Sandbox (isolado)	Menos isolamento
📚 Repositórios	Snap Store	Repositórios da distro
🖥️ Compatibilidade	Funciona em várias distros	Específico da distro

10. Snap

- A principais vantagens no uso do Snap são:

1 - Fácil de instalar (1 comando)

2 - Evita “dependência quebrada”

3 - Apps sempre atualizados

4 - Mais seguro (isolamento)

5 - Funciona em várias distribuições

10. Snap

- A principais desvantagens no uso do Snap são:

1 - Inicialização mais lenta (principalmente apps grandes)

2 - Consome mais espaço em disco

3 - Atualizações automáticas (pode incomodar)

4 - Integração com o sistema nem sempre perfeita

5 - Depende do serviço snapd rodando

6 – No geral, o APT é mais estável do que o snap

10. Snap

- O snap instala os programas em "/var/lib/snapd/". O quadro abaixo mostra os principais comandos do snap:

```
#instala o snap (no ubuntu já vem instalado)
sudo apt update
sudo apt install snapd

#habilita o serviço
sudo systemctl enable --now snapd

#verifica se está funcionando
snap version
```

10. Snap

- O quadro abaixo mostra os principais comandos do snap:

```
#instala VLC com o snap
sudo snap install vlc

#lista programas instalados com o snap
snap list

#remove um programa instalado com snap
sudo snap remove vlc

#atualiza todos os programas ou apenas um específico
sudo snap refresh
sudo snap refresh vlc
```

10. Snap

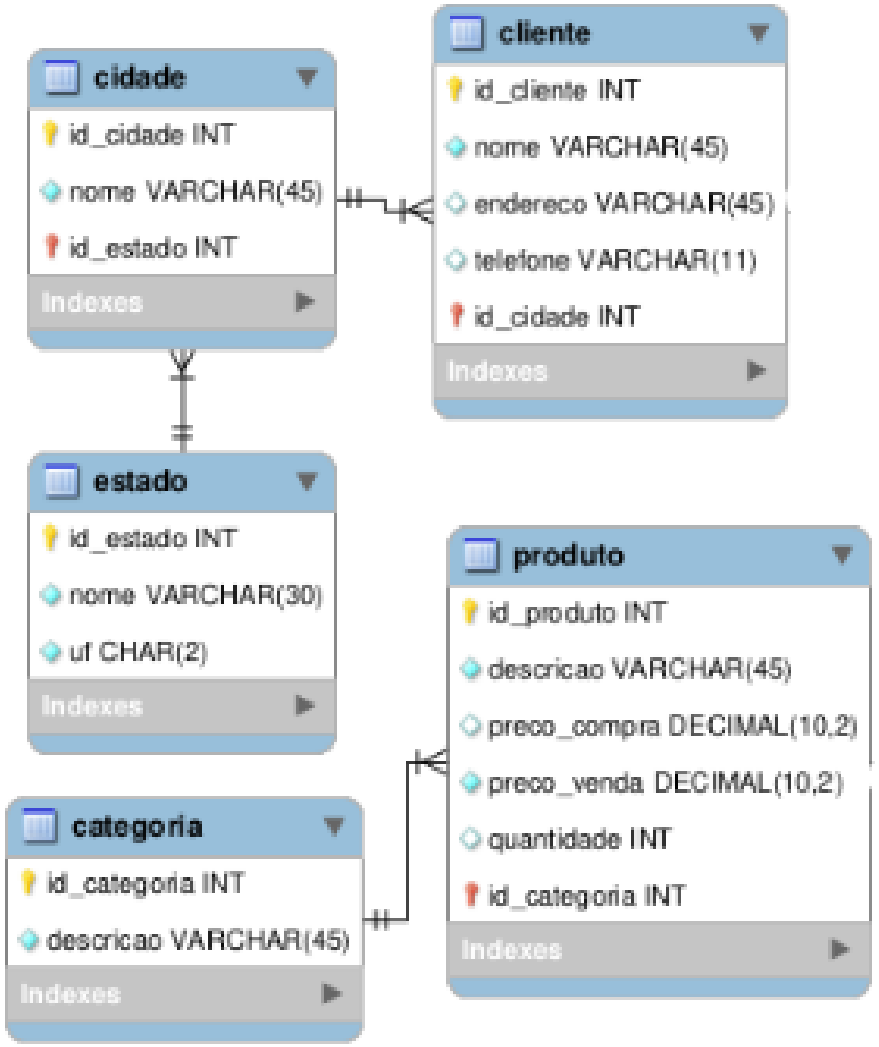
- Alguns programas precisam de acesso total ao sistema (não apenas no sandbox) para funcionar, como compiladores por exemplo. Neste caso, o snap deve ser instalado no modo clássico

Característica	Snap normal (sandbox)	Snap clássico
Acesso a arquivos	Limitado	Total
Segurança	Alta	Menor
Isolamento	Sim	Não
Integração com sistema	Limitada	Completa

```
#instala VLC com o snap
sudo snap install nome-do-app --classic
```

11. Vamos Brincar com o MySQL para ver se funciona?

- Vamos criar o seguinte banco:



11. Vamos Brincar com o MySQL para ver se funciona?

- Comandos básicos:

```
create database loja;

use loja
```

```
mysql –u root -p
```

```
CREATE TABLE `cliente` (
  `id_cliente` int PRIMARY KEY,
  `nome` varchar(50) NOT NULL,
  `endereco` varchar(50),
  `id_cidade` int(11) NOT NULL,
  `email` varchar(50)
);
```

```
ALTER TABLE cliente ADD (CONSTRAINT FK_Cidade
FOREIGN KEY(id_cidade) REFERENCES cidade(id_cidade)
);
```

11. Vamos Brincar com o MySQL para ver se funciona?

- Comandos básicos:

```
INSERT INTO `cliente` (`id_cliente`, `nome`, `endereco`, `id_cidade`, `telefone`) VALUES (NULL, 'Gabriel Jesus',
'Rua Juca, 45', '1', '44999566878');
```

```
UPDATE `cliente` SET `endereco` =  'Rua Tiradentes, 56',
`telefone` =  '4232316765',
`id_cidade` =  '2' WHERE `id_cliente`=2;
```

```
DELETE FROM `cliente` WHERE `id_cliente`=10;

Ou

DELETE FROM `cliente` WHERE `nome`='Felipe Melo';
```

11. Vamos Brincar com o MySQL para ver se funciona?

- Comandos básicos:

```
SELECT * FROM pedido WHERE data>='2017-01-01' AND data<='2017-12-31' AND total<=50000;
```

```
SELECT * FROM cliente ORDER BY nome DESC;
```